An Optimized Proportionate Adaptive Algorithm for Sparse System Identification - Project Report

K.R.Srinivas - EE18B136

This work is a part of *EE6110 - Adaptive Signal Processing* Course Project which involves reviewing, reproducing and analysing the results of the paper *"An optimized proportionate adaptive algorithm for sparse system identification"* by S. Ciochina, C. Paleologu, J. Benesty and S. L. Grant, 2015.

Abstract

For sparse system identification applications, proportionate type adaptive algorithms are popular due to their faster convergence and low misadjustment. This work proposes an optimized proportionate LMS algorithm in the context of time-varying system, minimizing the system misalignment. Simulations performed in the context of acoustic echo cancellation indicate the good features of the proposed algorithm.

I. Introduction

In several practical applications, such as network and acoustic echo cancellation the system impulse response to be estimated is sparse. Such impulse responses have small percentage of active coefficients (significant magnitude) and the rest are zero or small.

The sparseness character of the system is exploited to improve the conventional adaptive algorithms which are generally ignorant of the system sparsity. The idea is to proportionate the algorithm behavior, i.e., to update each coefficient of the filter independently of the others, by adjusting the adaptation step size in proportion to the magnitude of the estimated filter coefficient.The aim is to redistribute the adaptation gain proportionately, emphasizing larger ones, so that they would converge faster and also aid faster overall convergence.

One of the first kind of proportionate-type algorithm for system identification was Proportionate-NLMS (PNLMS) proposed by Duttweiler [1]. It was developed in an intutive manner and was sensitive to sparseness of the system impulse response, i.e., convergence significantly slows down if the system impulse response is dense. Later, the Improved-PNLMS [2] combined the usual NLMS update and proportionate factor in the same update rule. This algorithm was adaptable to any degree of sparseness decently. However, the equations used to calculate the step-size control factors are not based on any optimization criterion.

In the proposed work, a time-varying system following a first-order Markov model is assumed and proceed to minimize the system misalignment, a fair optimization criterion.Simulations performed in the context of acoustic echo cancellation (AEC) indicate that the proposed algorithm could be an attractive choice for sparse system identification problems. The algorithm is unique in the sense, it achieves faster convergence and tracking capability (to be able to dynamically model the long length and fast time-varying nature of the echo path impulse response.) along with lower misadjustment (to be able to attenuate the far-end speaker echoes) by setting proportionate factors depend on misalignment rather than magnitude of coefficients.

II. System Model

Let us consider the framework of a system identification problem (as shown in Fig. 1), like in AEC. The far end (or loudspeaker) signal, $x(n)$, goes through the echo path, $\mathbf{h}(n)$ (FIR of length M), providing the echo signal, y(n), where n is the time index. This signal is added to the near-end signal, $v(n)$ (which can contain both the background noise and the near-end speech), resulting the microphone signal, $d(n)$.

$$d(n) = \mathbf{x}^T(n)\mathbf{h}(n) + v(n), \tag{1}$$

where,

$$\mathbf{x}(n) = [x(n) \quad x(n-1) \quad \cdots \quad x(n-M+1)]^T$$
$$\mathbf{h}(n) = [h_0(n) \quad h_1(n) \quad \cdots \quad h_{M-1}(n)]^T$$

The near end signal in our case is considered as a zero-mean white Gaussian noise signal of variance $\sigma_v^2 = E[v(i)]^2$. The adaptive filter, defined by the vector $\hat{\mathbf{h}}(n)$ (FIR of length M),aims to produce at its output an estimate of the echo, $\hat{\mathbf{y}}(n)$, while the error signal, $e(n)$, should contain an estimate of the near-end signal. Note, both the adaptive filter and the echo path are

fed the same input far end signal.

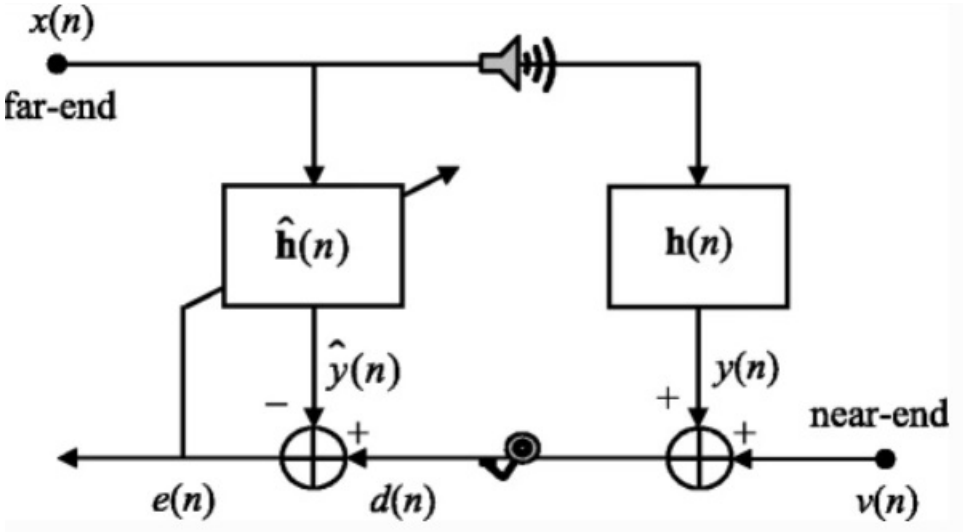

Fig. 1: Acoustic echo cancellation configuration

We assume The Impulse Response of the system to follow simplified first order Markov model,

$$\mathbf{h}(n) = \mathbf{h}(n-1) + \mathbf{w}(n), \tag{2}$$

where $\mathbf{w}(n)$ is a zero-mean white Gaussian noise signal vector, which is uncorrelated with $\mathbf{h}(n-1)$. The correlation matrix of $\mathbf{w}(n)$ is assumed to be $\mathbf{R}_w = \sigma_w^2 \mathbf{I}_M$, where $\mathbf{I}_M$ is the $M \times M$ identity matrix. The variance, σ_w^2 ,captures the uncertainties in $\mathbf{h}(n)$. Notice, equations (1) and (2) define a state variable model, similar to Kalman filtering.

III. An Optimized LMS-Based Algorithm with Individual Control Factors

The LMS based algorithm with individual control factors is defined by the update rule,

$$\widehat{\mathbf{h}}(n) = \widehat{\mathbf{h}}(n-1) + \mu \mathbf{K}(n-1)\mathbf{x}(n)e(n), \tag{3}$$

where $\mathbf{K}(n)$ is a diagonal matrix $(M \times M)$ containing the control factors at time index n, μ is the step-size of the algorithm and,

$$e(n) = d(n) - \mathbf{x}^T(n)\hat{\mathbf{h}}(n-1) \tag{4}$$

is the error signal of the adaptive filter.Also, let us define the a posteriori misalignment as $\mathbf{c}(n) = \mathbf{h}(n) - \hat{\mathbf{h}}(n)$, so that, developing (3) and (4), the update results in

$$\mathbf{c}(n) = \mathbf{c}(n-1) + \mathbf{w}(n) - \mu \mathbf{K}(n-1)\mathbf{x}(n)e(n). \tag{5}$$

$$e(n) = \mathbf{x}^T(n)\mathbf{c}(n-1) + \mathbf{x}^T(n)\mathbf{w}(n) + v(n). \tag{6}$$

We now proceed to minimize the system misalignment $\mathbf{c}(n)$, the optimization criterion after taking taking the l_2 norm in (5), then the mathematical expectations on both sides .

$$\begin{aligned} E\left[\|\mathbf{c}(n)\|_2^2\right] = E\left[\|\mathbf{c}(n-1)\|_2^2\right] + M\sigma_w^2 - 2\mu E\left[\mathbf{x}^T(n)\mathbf{K}(n-1)\mathbf{w}(n)e(n)\right] \\ - 2\mu E\left[\mathbf{x}^T(n)\mathbf{K}(n-1)\mathbf{c}(n-1)e(n)\right] + \mu^2 E\left[e^2(n)\mathbf{x}^T(n)\mathbf{K}^2(n-1)\mathbf{x}(n)\right]. \end{aligned} \tag{7}$$

Before attempting to minimize the system misalignment, we can simplify to great extent equation (7) by using equation (4), removing uncorrelated terms, assuming the input signal is white and the fact that $tr(AB) = tr(BA)$. The first cross-correlation term reduces to,

$$E\left[\mathbf{x}^T(n)\mathbf{K}(n-1)\mathbf{w}(n)e(n)\right] = E\left\{\text{tr}\left[\mathbf{w}(n)\mathbf{w}^T(n)\mathbf{K}(n-1)\mathbf{x}(n)\mathbf{x}^T(n)\right]\right\} = \sigma_w^2\sigma_x^2\text{tr}\left[\mathbf{K}(n-1)\right] \tag{8}$$

and the second reduces to,

$$E\left[\mathbf{x}^T(n)\mathbf{K}(n-1)\mathbf{c}(n-1)e(n)\right] = \text{tr}\left\{E\left[\mathbf{c}(n-1)\mathbf{c}^T(n-1)\right]E\left[\mathbf{x}(n)\mathbf{x}^T(n)\right]\mathbf{K}(n-1)\right\} = \sigma_x^2\text{tr}\left[\mathbf{R}_\mathbf{c}(n-1)\mathbf{K}(n-1)\right] \tag{9}$$

where $\mathbf{R}_\mathbf{c}(n) = E\left[\mathbf{c}(n)\mathbf{c}^T(n)\right]$. To simplify the third correlation term, we assume the input signal is stationary to some degree and that the filter tap-length is quite large, such that $||\mathbf{K}(n-1)\mathbf{x}(n)||_2^2$ is deterministic and use orthogonality principle assuming the algorithm has converged fairly.

$$E\left[e^2(n)\mathbf{x}^T(n)\mathbf{K}^2(n-1)\mathbf{x}(n)\right] = \text{tr}\left\{E\left[\mathbf{x}(n)\mathbf{x}^T(n)\right]\mathbf{K}^2(n-1)\right\}E\left[e^2(n)\right] = \sigma_x^2\text{tr}\left[\mathbf{K}^2(n-1)\right]\left[\sigma_v^2 + M\sigma_w^2\sigma_x^2 + \sigma_x^2 m(n-1)\right]. \tag{10}$$

where $m(n) = E\left[\|\mathbf{c}(n)\|_2^2\right]$. Finally, we can combine equations (8)-(10) and modify (7) as

$$m(n) = m(n-1) + M\sigma_w^2 - 2\mu\sigma_x^2\text{tr}\left\{\left[\mathbf{R}_\text{c}(n-1) + \sigma_w^2\mathbf{I}_M\right]\mathbf{K}(n-1)\right\} + \mu^2\sigma_x^2\text{tr}\left[\mathbf{K}^2(n-1)\right]\left[\sigma_v^2 + M\sigma_w^2\sigma_x^2 + \sigma_x^2 m(n-1)\right]. \tag{11}$$

Considering the step-size parameter to be time dependent, the system misalignment is minimized by setting $\frac{\partial m(n)}{\partial \mu(n)} = 0$ to obtain the step-size as,

$$\mu(n) = \frac{\text{tr}\left\{\left[\mathbf{R}_\mathbf{c}(n-1) + \sigma_w^2\mathbf{I}_M\right]\mathbf{K}(n-1)\right\}}{\text{tr}\left[\mathbf{K}^2(n-1)\right]\left\{\sigma_v^2 + \sigma_x^2[m(n-1) + M\sigma_w^2]\right\}}. \tag{12}$$

with the update equation of $m(n)$ modified as,

$$m(n) = m(n-1) + M\sigma_w^2 - \frac{\sigma_x^2\left\{\text{tr}\left\{\left[\mathbf{R}_\mathbf{c}(n-1) + \sigma_w^2\mathbf{I}_M\right]\mathbf{K}(n-1)\right\}\right\}^2}{\text{tr}\left[\mathbf{K}^2(n-1)\right]\left\{\sigma_v^2 + \sigma_x^2[m(n-1) + M\sigma_w^2]\right\}}. \tag{13}$$

We introduce the following new notations to simplify finding optimal solution.

$$\begin{aligned}
\mathbf{\Gamma}(n) &= \mathbf{R}_\mathbf{c}(n) + \sigma_w^2\mathbf{I}_M, \\
\boldsymbol{\gamma}(n) &= \left[\ \gamma_0(n) \quad \gamma_1(n) \quad \dots \quad \gamma_{M-1}(n)\ \right]^T, \\
\mathbf{k}(n) &= \left[\ k_0(n) \quad k_1(n) \quad \dots \quad k_{M-1}(n)\ \right]^T,
\end{aligned}$$

where $\gamma(n)$ and $\mathbf{k}(n)$ are two vectors containing the diagonal elements of $\Gamma(n)$ and $\mathbf{K}(n)$, respectively. Taking into account that $\mathbf{K}(n)$ is a diagonal matrix and based on the Cauchy- Schwarz inequality, we can write

$$\left\{\text{tr}\left[\mathbf{\Gamma}(n-1)\mathbf{K}(n-1)\right]\right\}^2 \leq \|\boldsymbol{\gamma}(n-1)\|_2^2\|\mathbf{k}(n-1)\|_2^2. \tag{14}$$

where the equality is obtained iff $\mathbf{k}(n-1)$ and $\gamma(n-1)$ are proportional to each other, i.e, $\mathbf{k}(n-1) = q\gamma(n-1)$ with $p > 0$. This can be equivalently described as,

$$\begin{aligned}
&\text{tr}\ \left[\mathbf{K}(n-1)\right] = q\text{tr}\left[\mathbf{\Gamma}(n-1)\right] = q\left[m(n-1) + M\sigma_w^2\right], \\
&\text{tr}\ \left[\mathbf{K}^2(n-1)\right] = \|\mathbf{k}(n-1)\|_2^2 = q^2\|\boldsymbol{\gamma}(n-1)\|_2^2, \\
&\text{tr}\ \left[\mathbf{\Gamma}(n-1)\mathbf{K}(n-1)\right] = q\|\boldsymbol{\gamma}(n-1)\|_2^2
\end{aligned} \tag{15}$$

For simplicity we could impose $\text{tr}\left[\mathbf{K}(n-1)\right] = M$, as it is the case with conventional non-proportionate algorithms (where $\mathbf{K}(n-1) = I_M$), except the gains will be individually distributed through the elements of $\gamma(n-1)$. Using first relation of equation (15) we get,

$$q = \frac{M}{m(n-1) + M\sigma_w^2}. \tag{16}$$

Consequently equation (12),(13) (3) can be simplified to the following recursive equations.

$$\mu(n) = \frac{1}{q\left\{\sigma_v^2 + \sigma_x^2\left[m(n-1) + M\sigma_w^2\right]\right\}} = \frac{1}{q\sigma_v^2 + M\sigma_x^2} \tag{17}$$

$$m(n) = m(n-1) + M\sigma_w^2 - q\mu(n)\sigma_x^2\|\boldsymbol{\gamma}(n-1)\|_2^2. \tag{18}$$

$$\widehat{\mathbf{h}}(n) = \widehat{\mathbf{h}}(n-1) + q\mu(n)\boldsymbol{\gamma}(n-1) \odot \mathbf{x}(n)\mathbf{e}(n), \tag{19}$$

In order to evaluate $\gamma(n-1)$ similarly, we notice that the diagonal elements of $\Gamma(n-1)$ depends on $\mathbf{R_c}(n-1)$, which in turn can be solved iteratively using relation (5) in $\mathbf{R_c}(n) = E\left[\mathbf{c}(n)\mathbf{c}^T(n)\right]$. Thus,

$$\begin{aligned}\mathbf{R_c}(n) &= \mathbf{R_c}(n-1) + \sigma_w^2\mathbf{I}_M - \mu E\left[\mathbf{w}(n)\mathbf{x}^T(n)\mathbf{K}(n-1)e(n)\right] \\ &- \mu E\left[\mathbf{K}(n-1)\mathbf{x}(n)\mathbf{w}^T(n)e(n)\right] - \mu E\left[\mathbf{c}(n-1)\mathbf{x}^T(n)\mathbf{K}(n-1)e(n)\right] \\ &- \mu E\left[\mathbf{K}(n-1)\mathbf{x}(n)\mathbf{c}^T(n-1)e(n)\right] + \mu^2 E\left[e^2(n)\mathbf{K}(n-1)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{K}(n-1)\right].\end{aligned} \tag{20}$$

which can be solved similarly to system misalignment done before and finally obtain,

$$\gamma(n) = \gamma(n-1) + \sigma_w^2 1_{M\times 1} + q\mu(n)\sigma_x^2\left\{\sigma_v^2 + \sigma_x^2\left[m(n-1) + M\sigma_w^2\right] - 2q\mu(n)\right\} \times \gamma(n-1) \odot \gamma(n-1),$$

$$\gamma(n) = \gamma(n-1) + \sigma_w^2 1_{M\times 1} + \sigma_x^2\left\{1 - 2q^2\mu^2(n)\right\} \times \gamma(n-1) \odot \gamma(n-1), \tag{21}$$

where $\mathbf{1}_{M\times 1}$ denotes a column vector with all its L elements equal to one. Due to stability reasons, a normalization should be also performed in this step, i.e., $\bar{\gamma}_i(n) = \gamma_i(n)/max[q\gamma(n)]$, with $0 \leq i \leq M-1$. Summarizing, the resulting optimized LMS algorithm with individual control factors is defined by the relations (16)–(19) and (21) (including the normalization).

IV. Simulation Results

As mentioned before simulations are performed in the context of Acoustic Echo Cancellation (Figure 1). In this case, the adaptive filter is used to identify the acoustic echo path between the loudspeaker and the microphone,i.e., the room impulse response.The main challenges of this application include the high-length and time-varying nature of the acoustic impulse response, as well as the non-stationary character of the input signal. The sparseness of an acoustic impulse response is more problematic because it depends on many factors, e.g., reverberation time, the distance between loudspeaker and microphone, different changes in the environment (e.g., temperature or pressure). Hence, we consider two impulse responses, one of them being quite sparse and the other being dense. Each of them have 512 coefficients with 8kHz sampling rate [3]. The length of the adaptive filter is usually less than impulse response length (in practical AEC scenarios), however for convenience we set it to $M = 512$ as well.

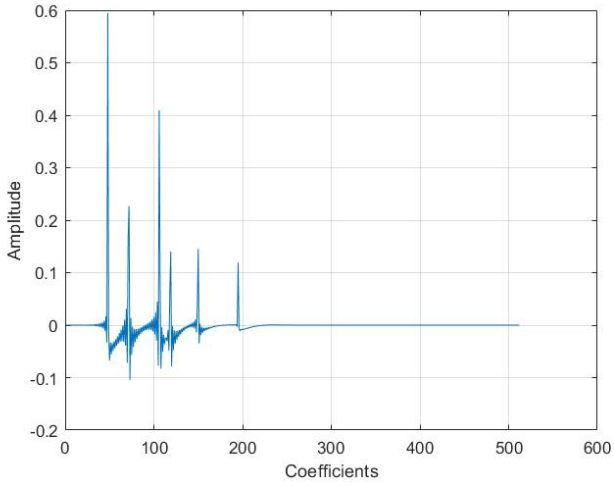

Fig. 2: Sparse Impulse Response used in simulation

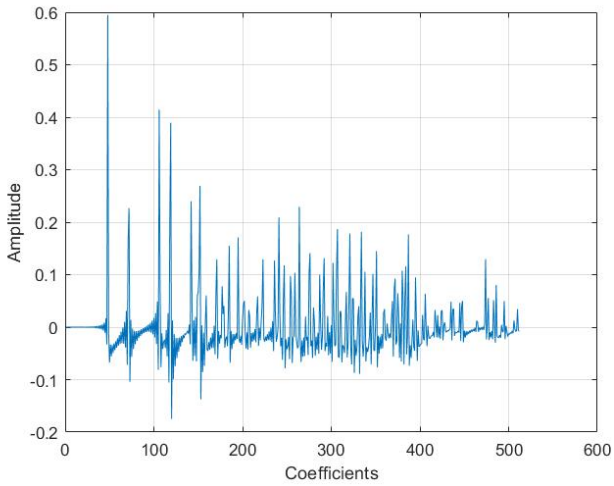

Fig. 3: Dense Impulse Response used in simulation

The far end signal is either a unit variance AR(1) process generated by filtering a unit variance white gaussian noise through a first order system $\sqrt{1-a^2}/(1-az^{-1})$, with $a = 0.8$ or a speech sequence. The echo signal is corrupted with white gaussian noise signal (near end) with 20dB SNR (Note: $\sigma_v^2 = 0.01$ since $\sigma_x^2 = 1$) .In order to evaluate the tracking capabilities of the algorithms, an echo path change scenario is simulated in some simulations, by shifting the impulse response to the right by 50 samples. The measure of performance is the normalized misalignment (in dB), defined by $20log_{10}||\mathbf{h}(n)-\widehat{\mathbf{h}}(n)||_2/||\mathbf{h}(n)||_2$.

Optimized Proportionate LMS Algorithm

Initialization: $\hat{\mathbf{h}}(0) = 0 \quad \hat{m}(0) = \epsilon > 0 \quad \hat{\sigma}_w^2 = 0 \quad \gamma(0) = 1_{M\times 1}$

Iteration :

. $e(n) = d(n) - \mathbf{x}^T(n)\hat{\mathbf{h}}(n-1),$

. $\hat{\sigma}_x^2 = \frac{1}{M}\mathbf{x}^T(n)\mathbf{x}(n),$

. $q = M/(m(n-1) + M\hat{\sigma}_w^2)$

. $\mu(n) = \frac{1}{q\sigma_v^2 + M\sigma_x^2},$

. $m(n) = m(n-1) + M\sigma_w^2 - q\mu(n)\sigma_x^2\|\boldsymbol{\gamma}(n-1)\|_2^2,$

. $\widehat{\mathbf{h}}(n) = \widehat{\mathbf{h}}(n-1) + q\mu(n)\boldsymbol{\gamma}(n-1) \odot \mathbf{x}(n)\mathbf{e}(n),$

. $\gamma(n) = \gamma(n-1) + \sigma_w^2 1_{M\times 1} + \sigma_x^2\left\{1 - 2q^2\mu^2(n)\right\} \times \gamma(n-1) \odot \gamma(n-1),$

. $\gamma(n) = \gamma(n)/max(q\gamma(n))$

. $\hat{\sigma}_w^2 = \frac{1}{M}||\widehat{\mathbf{h}}(n) - \widehat{\mathbf{h}}(n-1)||^2$

Improved Proportionate LMS Algorithm Refer [2]

Initialization: $\hat{\mathbf{h}}(0) = 0$

Parameters: Set appropriate μ, δ and $-1 \leq \alpha < 1$

Iteration :

. $e(n) = d(n) - \mathbf{x}^T(n)\hat{\mathbf{h}}(n-1),$

. $k_i(n) = \frac{1-\alpha}{2M} + (1+\alpha)\frac{||h_i(n-1)||_1}{2||\hat{\mathbf{h}}(n-1)||_1+\delta}$

. $Q = \text{diag}\{\mathbf{k}(n)\}$

. $\widehat{\mathbf{h}}(n) = \widehat{\mathbf{h}}(n-1) + \frac{\mu Q\mathbf{x}(n)\mathbf{e}(n)}{\mathbf{x}^T(n)Q\mathbf{x}(n)+\epsilon},$

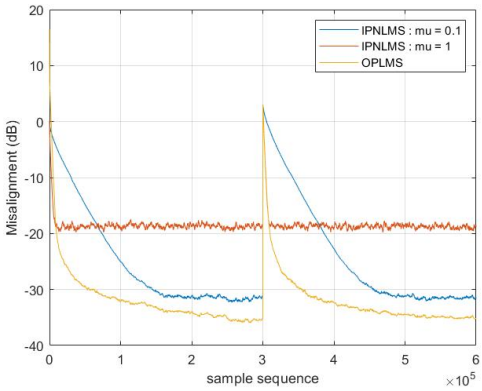

Fig. 4: Misalignment of the IPNLMS (with different values of μ) and OPLMS algorithms for Sparse Acoustic impulse response from Fig(2). echo path changes after 3×10^5 input samples. The input signal is an AR1(0.8) process.

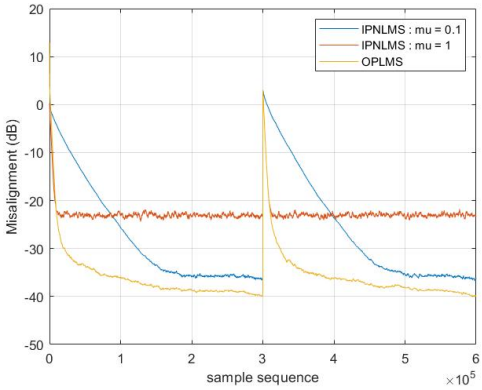

Fig. 5: Misalignment of the IPNLMS (with different values of μ) and OPLMS algorithms for Dense Acoustic impulse response from Fig(2). echo path changes after 3×10^5 input samples. The input signal is an AR1(0.8) process.

The main goal of the simulation is to highlight the variable step size feature of the OPLMS algorithm. For this we compare with IPNLMS [2] at two different μ values. For $\mu = 1$ (fastest convergence mode), both the algorithms have almost same initial convergence. However, the IPNLMS achieves lower (poor) steady-state misalignment. On the other hand, for $\mu = 0.1$, in steady-state (provided echo path doesn't change), both algorithm tend to achieve almost same misalignment but IPNLMS has poor convergence rate. These conclusions are valid irrespective of sparseness of the impulse response.

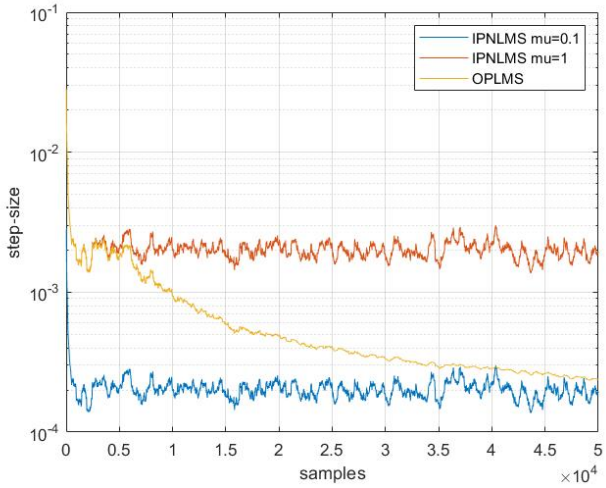

Fig. 6: Comparison of step size evolution in IPNLMS and OPLMS filters.

In the above plot (Fig 6) we compare the maximum OPLMS filter coefficient with that of IPNLMS filter coefficient for μ = 0.1 and 1. Initially, the stepsize of OPLMS is quite high and comparable to IPNLMS (μ = 1) aiding faster convergence. But later, it decreases gradually on achieving steady-state, comparable to IPNLMS ($\mu = 0.1$) where their misalignments are also similar. This feature of OPLMS filter helps it achieve faster convergence rate.

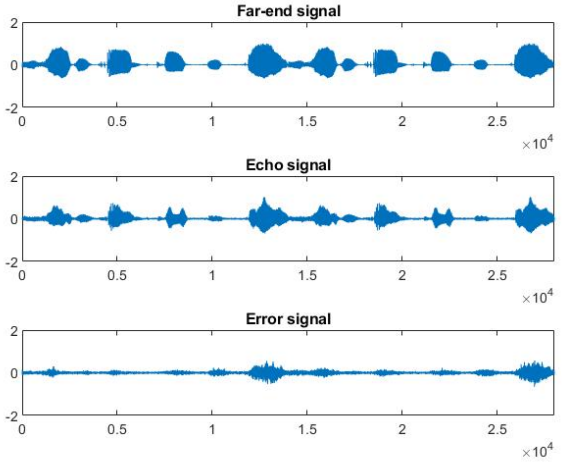

Fig. 7: The top row shows the far end signal, the middle row shows the corresponding echo and the last row shows the error signal resulting from IPNLMS ($\mu = 1$) filter

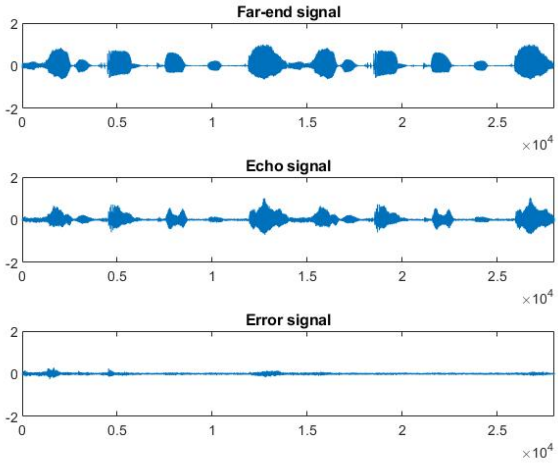

Fig. 8: The top row shows the far end signal, the middle row shows the corresponding echo and the last row shows the error signal resulting from OPLMS filter

We now use real speech sequence (of length N = 28017) to visualise the attenuation of echo on using the adaptive filter. It is quite evident from the simulation results that the strength of error signal resulting in OPLMS filter weaker than that of IPNLMS filter

V. Conclusion

Thus the proposed Optimized Proportionate LMS for a time-variant system model, assuming optimization criterion based on misalignment promise to be a better alternative to conventional as well as other proportionate-type algorithms in terms of convergence, tracking ability and misadjustment. The simulation results support our arguments by showing how robust is the proposed algorithm to variations in system impulse response.

References

[1] D. L. Duttweiler, "Proportionate normalized least-mean-squares adaptation in echo cancelers," in IEEE Transactions on Speech and Audio Processing, vol. 8, no. 5, pp. 508-518, Sept. 2000, doi: 10.1109/89.861368.

[2] J. Benesty and S. L. Gay, "An improved PNLMS algorithm," in Proc. IEEE ICASSP, 2002, pp. II-1881–II-1884.

[3] J.B. Allen and D.A. Berkley, "Image method for efficiently simulating small-room acoustics," Journal Acoustic Society of America, 65(4), April 1979, p 943. www.audiolabs-erlangen.de/fau/professor/habets/software/rir-generator

```
%% EE6110 Project: Author - K.R.SRINIVAS EE18B136

% Code A : The below program allows you plot and compare the misalignment
% for different algorithms.

clear all
close all

load IR_sparse.mat;  % loads the echo path model
load speech.mat ;    % loads the far end signal; can use if synthetic signal not reqd

%% Genaration of synthetic speech sequence

a = 0.8 ;
Nr = [sqrt(1-a^2)] ;
Dr = [1 -a] ;
N = 600000 ;                  % sequence length
u = randn(N,1);
far_end = filter(Nr,Dr,u);    % input sequence of unit variance (AR(1) process)
var_n = 0.01 ;                % Near end Background noise
N = length(far_end);

%% Impulse Response of the system

h1 = IR_sparse ;
h2 = circshift(h1,50) ;       % shifted impulse response to model change in echo path model

%% Echo/Desired signal

echo_1 = filter(h1,1,far_end(1:N/2)) ;
echo_2 = filter(h2,1,far_end((N/2)+1:end)) ;
echo = [echo_1;echo_2] + sqrt(var_n)*randn(N,1) ;  % desired signal (echo + near end)

%% Initializations
M = 512 ;                     % Filter-Tap Length

% IPNLMS Specifications mu = 0.1
mu = 0.1 ;
alpha = 0 ;
delta = 9.8314e-04 ;
epsilon = 0.01 ;

w0 = zeros(M,1) ;             % Weight vector
u0 = zeros(M,1) ;             % regressor vector
m0 = zeros(N,1) ;             % misalignment vector
```

```
disp('Please wait for a while...')

%% IPNLMS Algorithm

for i=1:N
    u0 = [far_end(i);u0(1:M-1)];   % Regressor vector update
    e0(i) = echo(i) - u0'*w0;      % apriori error

    for s = 1:M    % proportionate step-size implementation
        k(s) = (1-alpha)/2*M + (1+alpha)*norm(w0(s),1)/(2*norm(w0,1)+delta) ;
    end

    Q = diag(k) ;                  % Step-Size update matrix

    w0 = w0 + (mu*e0(i)*Q*u0)/(u0'*Q*u0 + epsilon) ;    % weight update rule

    if i <= N/2                                         % misalignment
         m0(i) = 20*log10(norm(h1-w0)/norm(h1)) ;
    else
         m0(i) = 20*log10(norm(h2-w0)/norm(h2)) ;
    end

    if mod(i,5000)==0
         i
    end
end

% IPNLMS Specifications  mu = 1
mu = 1 ;
alpha = 0 ;
delta = 9.8314e-04 ;
epsilon = 0.01 ;

w3 = zeros(M,1) ;            % Weight vector
u3 = zeros(M,1) ;            % regressor vector
m3 = zeros(N,1) ;            % misalignment vector


disp('Please wait for a while...')
%% IPNLMS Algorithm

for i=1:N
    u3 = [far_end(i);u3(1:M-1)];   % Regressor vector update
    e3(i) = echo(i) - u3'*w3;      % apriori error
```

```
    for s = 1:M     % proportionate step-size implementation
        k(s) = (1-alpha)/2*M + (1+alpha)*norm(w3(s),1)/(2*norm(w3,1)+delta) ;
    end

    Q = diag(k) ;                   % Step-Size update matrix

    w3 = w3 + (mu*e3(i)*Q*u3)/(u3'*Q*u3 + epsilon) ;   % weight update rule

    if i <= N/2                                        % misalignment
         m3(i) = 20*log10(norm(h1-w3)/norm(h1)) ;
    else
         m3(i) = 20*log10(norm(h2-w3)/norm(h2)) ;
    end

    if mod(i,5000)==0
         i
    end
end

% OPLMS Specifications

w2 = zeros(M,1) ;             % Weight vector
u2 = zeros(M,1) ;             % regressor vector
m2 = zeros(N,1) ;             % misalignment matrix
m = 1e-2 ;
var_w = 0 ;                   % process noise
gamma = ones(M,1);
I_1 = ones(M,1)  ;

%% OPLMS Algorithm

for i = 1:N
   u2 = [far_end(i) ; u2(1:M-1)] ;          % Regressor vector update
   e2(i) = echo(i)-u2'*w2 ;                 % apriori error

   var_x = (u2'*u2)/M  ;                    % variance of input signal
   q = M /(m + M*var_w);
   mu_new = 1/(q*var_n + var_x*M) ;
   w2_old = w2 ;
   w2 = w2 + q*mu_new*(gamma.*u2)*e2(i) ;  % weight vector update rule
   p = gamma ;
   gamma = gamma + var_w*I_1 + q*mu_new*var_x*(var_n+var_x*(m+M*var_w)-...
           2*q*mu_new)*(gamma.*gamma) ;
   r = max(q*gamma) ;
   gamma = (1/r)*gamma ;
   m = m + M*var_w - q*mu_new*var_x*norm(p)^2 ;
```

```
   var_w = (1/M)*norm(w2-w2_old)^2 ;      % process noise variance update


   if i <= N/2                            % misalignment
         m2(i) = 20*log10(norm(h1-w2)/norm(h1)) ;
   else
         m2(i) = 20*log10(norm(h2-w2)/norm(h2)) ;
   end

   if mod(i,5000)==0
        i
   end
end

plot(1:N,m0); hold on
plot(1:N,m3); hold on
plot(1:N,m2);
xlabel('sample sequence');
ylabel('Misalignment (dB)');
legend('IPNLMS : mu = 0.1','IPNLMS : mu = 1','OPLMS');
grid

% Code B: The below program allows you analyse the evolution of step-size in
% the update rule for OPLMS filter in comparison with IPNLMS filter.

clear all
close all

load IR_sparse.mat;   % loads the echo path model
load speech.mat  % loads a speech signal

%% Genaration of synthetic speech sequence

a = 0.8 ;
Nr = [sqrt(1-a^2)] ;
Dr = [1 -a] ;
N = 50000 ;                 % sequence length
u = randn(N,1);
far_end = filter(Nr,Dr,u);  % input sequence of unit variance (AR(1) process)
var_n = var(far_end)/100 ;  % Near end Background noise
N = length(far_end) ;

%% Impulse Response of the system

ho = IR_sparse ;
```

```
% ho = path ;

%% Echo/Desired signal

echo = filter(ho,1,far_end) ;
echo = echo + sqrt(var_n)*randn(N,1) ;

%% Initializations
M = length(ho) ;                    % Filter-Tap Length

% IPNLMS Specifications
mu = 0.1 ;
alpha = 0 ;
delta = 9.8314e-04 ;
epsilon = 0.1 ;

w0 = zeros(M,1) ;             % Weight vector
u0 = zeros(M,1) ;             % regressor vector
m0 = zeros(N,1) ;             % misalignment vector


disp('Please wait for a while...')
% IPNLMS Algorithm

for i=1:N
    u0 = [far_end(i);u0(1:M-1)];
    e0(i) = echo(i) - u0'*w0;

    for s = 1:M               % proportionate step-size implementation
        k(s) = (1-alpha)/2*M + (1+alpha)*norm(w0(s),1)/(2*norm(w0,1)+delta) ;
    end

    Q = diag(k) ;                   % Step-Size update matrix

    max_coeff0(i) = max(abs(mu*k/(u0'*Q*u0 + epsilon))) ;
    min_coeff0(i) = min(abs(mu*k/(u0'*Q*u0 + epsilon))) ;

    w0 = w0 + (mu*e0(i)*Q*u0)/(u0'*Q*u0 + epsilon) ;

    m0(i) = 20*log10(norm(ho-w0)/norm(ho)) ;

    if mod(i,5000)==0
         i
    end
end
```

```
% IPNLMS Specifications
mu = 1 ;
alpha = 0 ;
delta = 9.8314e-04 ;
epsilon = 0.01 ;

w3 = zeros(M,1) ;            % Weight vector
u3 = zeros(M,1) ;            % regressor vector
m3 = zeros(N,1) ;            % misalignment vector


disp('Please wait for a while...')
%% IPNLMS Algorithm

for i=1:N
    u3 = [far_end(i);u3(1:M-1)];
    e3(i) = echo(i) - u3'*w3;

    for s = 1:M              % proportionate step-size implementation
        k(s) = (1-alpha)/2*M + (1+alpha)*norm(w3(s),1)/(2*norm(w3,1)+delta) ;
    end

    Q = diag(k) ;                  % Step-Size update matrix

    max_coeff2(i) = max(mu*k/(u3'*Q*u3 + epsilon)) ;
    min_coeff2(i) = min(mu*k/(u3'*Q*u3 + epsilon)) ;

    w3 = w3 + (mu*e3(i)*Q*u3)/(u3'*Q*u3 + epsilon) ;

    m3(i) = 20*log10(norm(ho-w3)/norm(ho)) ;

    if mod(i,5000)==0
         i
    end
end



% OPLMS Specifications

w2 = zeros(M,1) ;            % Weight vector
u2 = zeros(M,1) ;            % regressor vector
m2 = zeros(N,1) ;            % misalignment matrix
m = 1e-2 ;
var_w = 0 ;                  % process noise
gamma = ones(M,1);
```

```
I_l = ones(M,1)  ;

%% OPLMS Algorithm

for i = 1:N
   u2 = [far_end(i) ; u2(1:M-1)] ;
   e2(i) = echo(i)-u2'*w2 ;

   var_x = (u2'*u2)/M  ;
   q = M /(m + M*var_w);
   mu_new = 1/(q*(var_n+var_x*(m+M*var_w))) ;

   w2_old = w2 ;
   w2 = w2 + q*mu_new*(gamma.*u2)*e2(i) ;

   max_coeff1(i) = max(abs(q*mu_new*gamma)) ;
   min_coeff1(i) = min(abs(q*mu_new*gamma)) ;

   m = m + M*var_w - q*mu_new*var_x*norm(gamma,2)^2 ;

   gamma = gamma + var_w*I_l + q*mu_new*var_x*(var_n+var_x*(m+M*var_w)-...
           2*q*mu_new)*(gamma.*gamma) ;
   r = max(q*gamma) ;
   gamma = (1/r)*gamma ;

   var_w = (1/M)*norm(w2-w2_old,2)^2 ;


   m2(i) = 20*log10(norm(ho-w2,2)/norm(ho,2)) ;

   if mod(i,5000)==0
        i
   end
end


semilogy(20:N,max_coeff0(20:N)); hold on
semilogy(20:N,max_coeff2(20:N)); hold on
semilogy(20:N,max_coeff1(20:N));
xlabel('samples');
ylabel('step-size');
legend('IPNLMS mu=0.1','IPNLMS mu=1','OPLMS') ;
grid
```